# Exam – My Money

You need to create a **web application** where users **see and manage personal finance**. Usually the users will visit the home-page, navigate either to **Registration or Login**.

**After login** they go to **All Operations** page, where all **operations are listed**. Then they can either **add** a new operation, **edit** or **delete** existing, **or see the Statistics** of all existing operations.

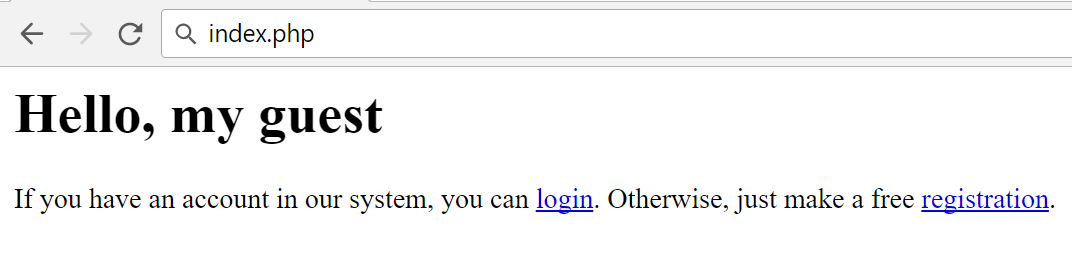
## Expected Result

General rules:

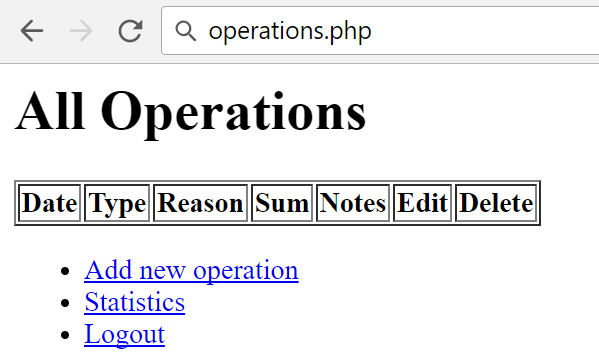
1. **No blank lines** should be left
2. Going **on new line** (click ENTER) in the same text should be done, **only after the sentence ends** (after full stop)
3. Each **title** is **Heading 1**
4. All **attribute values and tags** are **lowercase**
5. All name fields are lower\_snake\_case variant of the text/value
6. All **tables** have **head** and if they are filled with data, they have a **body**
7. Only **direct child tags** go on **new line** e.g. <tr> **is** on new line after <thead>, but <a> in <li> **is not** on new line.
8. 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.
9. All **error messages** are visualized on the **next line after the title** in a **paragraph** with style color=red.
10. All **relational dropdowns** follow the convention: name\_snake\_case\_id

**index.php:**

* When user **is** **not** logged ( <https://judge.softuni.bg/Contests/Compete/Index/845#0> (1 point)):

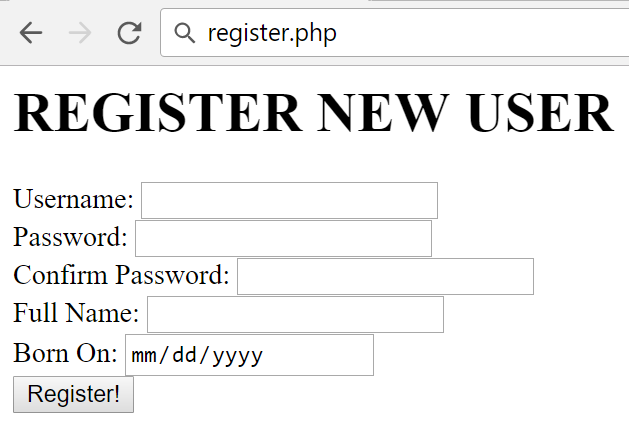


* When user **is** logged ( <https://judge.softuni.bg/Contests/Compete/Index/845#1> (1 point)):

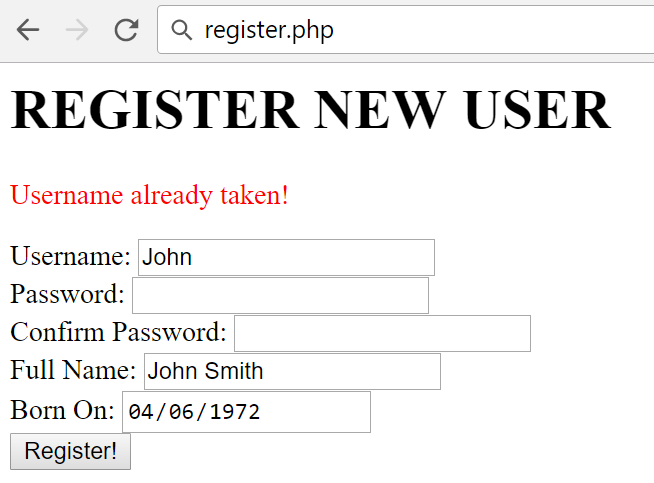


**register.php:**

* **Before** form is submitted ( <https://judge.softuni.bg/Contests/Compete/Index/845#2> (1 point)):



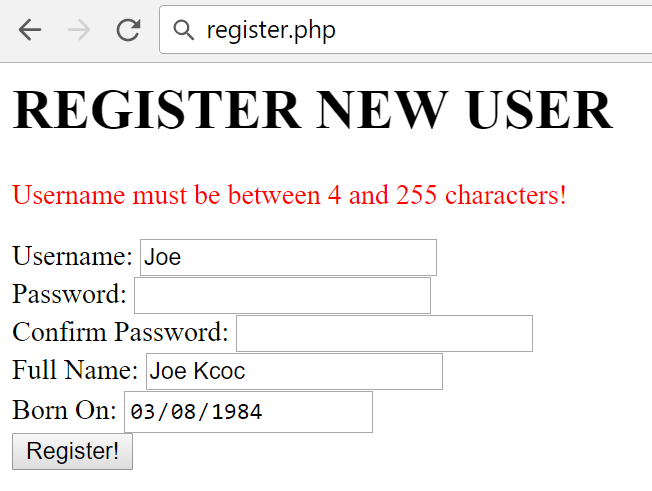
* Form **is submitted**, but there are **username** **duplicates (** <https://judge.softuni.bg/Contests/Compete/Index/845#3> **(1 point))**:



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

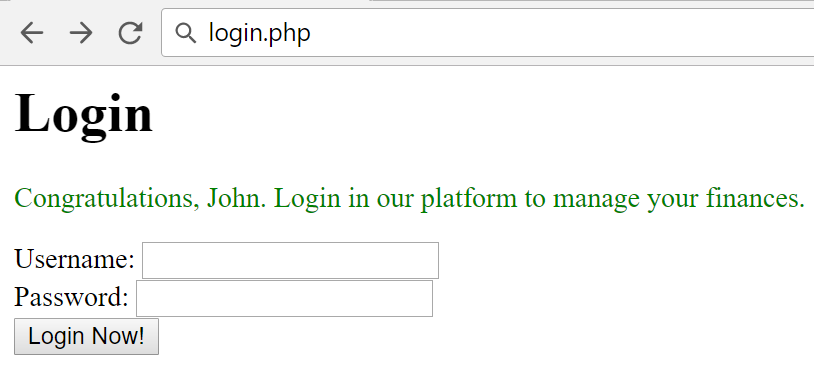


* Form **is submitted**, but **validation** did not pass:



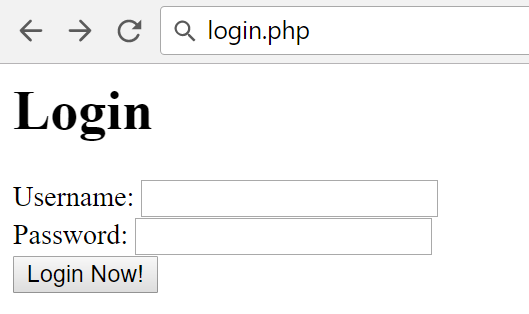
Show the **error only for the first field the validation did not pass** (they might all be out of range). The message is in the format:

* + 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 ( <https://judge.softuni.bg/Contests/Compete/Index/845#4> (1 point)):

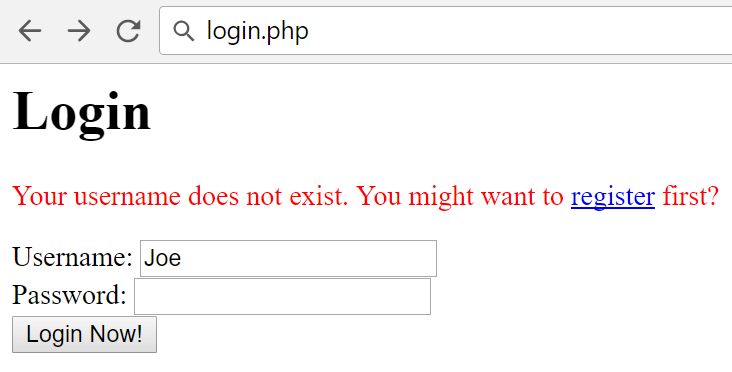


**login.php:**

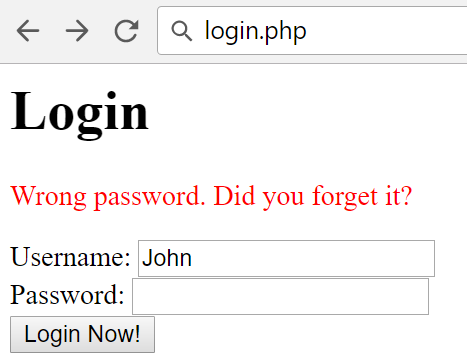
* **Before** form is submitted ( <https://judge.softuni.bg/Contests/Compete/Index/845#5> (1 point)):



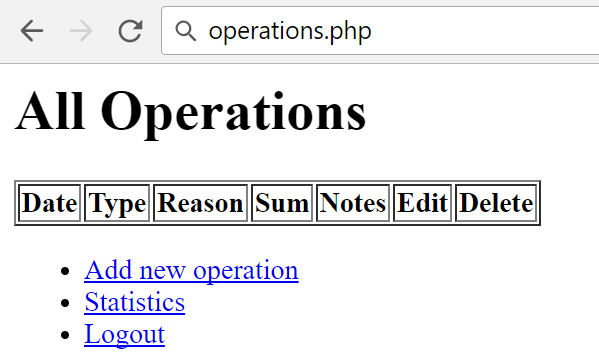
* Form **is submitted**, but **username** does not exist:



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



* **After** successful login:



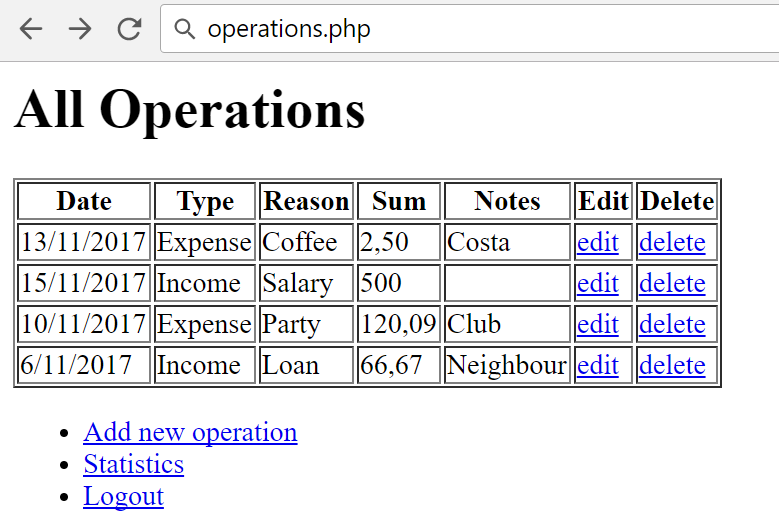
If there are operations, **list** them sorted by for\_date the following way:

* From the **nearest future date to the farthest**, then if there are **dates in the past, from the nearest to the farthest**. E.g. if today is 12 November 2017, and there are the following dates:
  + 10 November 2017
  + 15 November 2017
  + 6 November 2017
  + 13 November 2017

You need to order them:

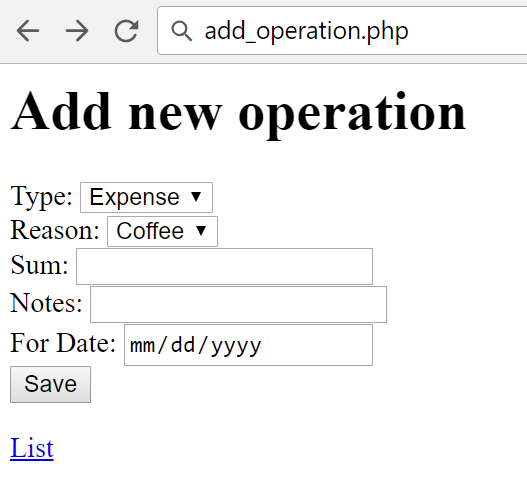
* + 13 November 2017
  + 15 November 2017
  + 10 November 2017
  + 6 November 2017

If there are **two or more equal dates**, order them by **primary key** of the operation ( <https://judge.softuni.bg/Contests/Compete/Index/845#6> (1 point)).

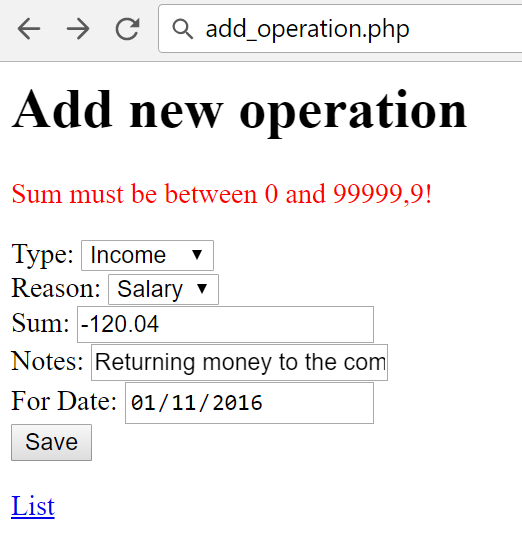


**add\_operation.php** (Add new operation):

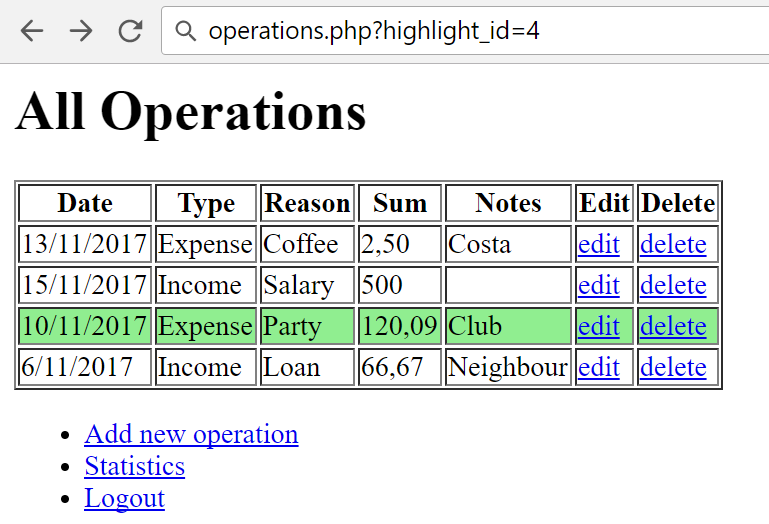
* **Before** form is submitted:



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

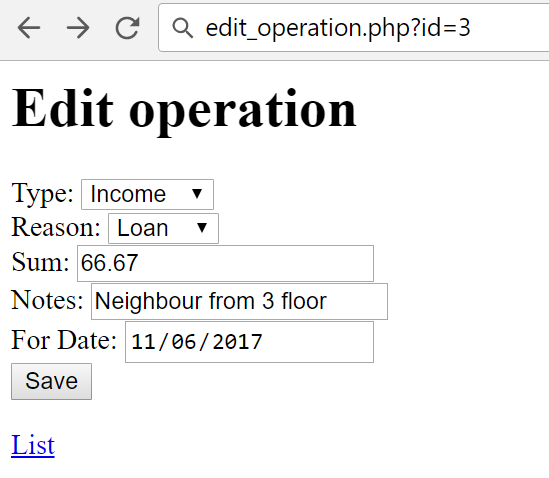


* After **successful submission**, the operation is **added**, **redirected** to the list of operations, and the row with new operation is **highlighted** with style background-color:lightgreen:

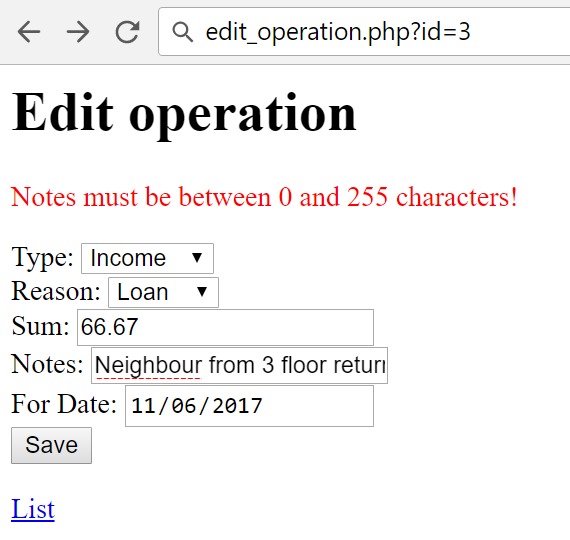


**edit\_operation.php?id={operation\_id}** (Edit existing operation):

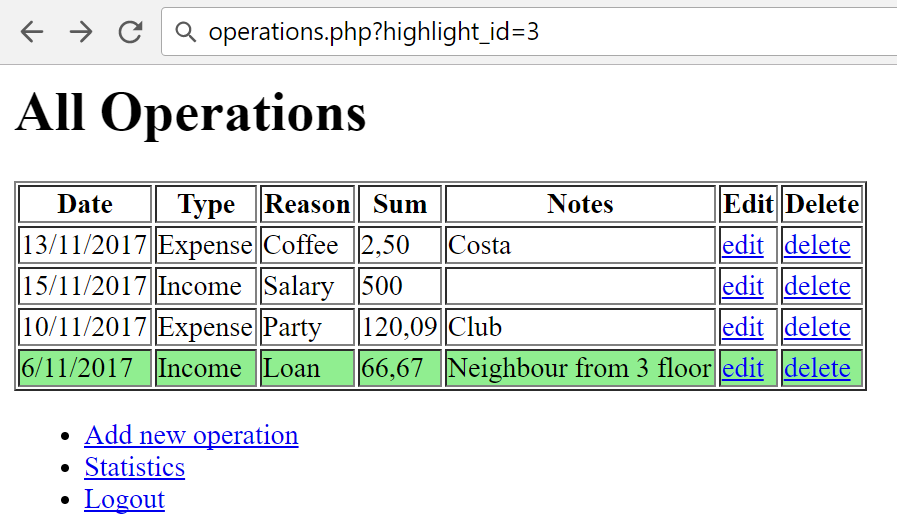
* **Before** form is submitted:



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



* After **successful submission**, the operation is **edited**, **redirected** to the list of operations, and the row with editted operation is **highlighted** with style background-color:lightgreen ( <https://judge.softuni.bg/Contests/Compete/Index/845#7> (1 point)):

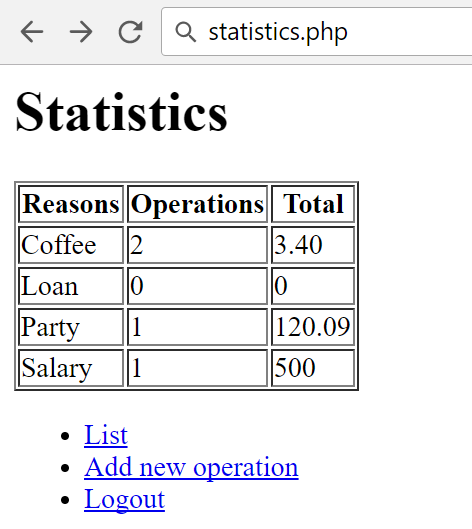


**delete\_operation.php?id={operation\_id}** (Delete existing operation):

**Deletes** the operation and **redirects** to operations list again.

**statistics.php** (Report for operation reasons):

Lists all **reasons, the operation count and operation total sum**, ordered by the **reason name**:



## Database

Create a database “**money**” and add the following tables:

### Data Definition

Submit in judge the **create table statements** as follows:

### DDL – Users (15 points)

|  |  |  |
| --- | --- | --- |
| **users** | | |
| **Column Name** | **Type** | **Constraints** |
| username | String | 4…255. Unique. |
| password | String | 4…255 |
| full\_name | String | 5…255 |
| born\_on | date | not null |

### DDL – Operations (25 points)

|  |  |  |
| --- | --- | --- |
| **operations** | | |
| **Column Name** | **Type** | **Constraints** |
| type | char | 1...1 |
| reason\_id | int | 1...4294967295 |
| sum | decimal | 0...99999,99 |
| notes | string | 0...255 |
| on\_date | Date Time | not null, current time |
| for\_date | Date | possible null |
| user\_id | int | 1...4294967295 |

### DDL – Reasons (10 points)

|  |  |  |
| --- | --- | --- |
| **reasons** | | |
| **Column Name** | **Type** | **Constraints** |
| name | String | 3…100, unique |

### DDL – Relations (50 points)

Add the needed relations to the above tables and paste in judge all CREATE TABLE definitions at once, this time with relations.

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

### Data Manipulation

Submit in judge **queries for adding or taking** data as follows:

### DML – Add Reasons (5 points)

Submit a query (or queries) which **adds** the following **reasons**:

|  |
| --- |
| **name** |
| Salary |
| Coffee |
| Party |
| Loan |

### DML – Add Users (10 points)

Submit a query (or queries) which **adds** the following **users**:

|  |  |  |  |
| --- | --- | --- | --- |
| **username** | **password** | **full\_name** | **born\_on** |
| peter | $2y$10$fnncUQdY0MQJuqIgFk1BGOQKkxpSmPXTh0IX.av4TB5Tp9CzM2PS. | Peter Jakson | 1/1/1952 |
| James | $2y$10$Q7A5iTVJBYnaY5X5gMwqOuoD5pGcDooG9c7k5PqM2.0L9LRSmDX7K | James Cameron | 4/12/1945 |
| Alice | $2y$10$qh.HcdLKZwsnl4GglZ43h.Cti5Q9WGPYjnTbJu9Ad28e3jZWRFv5G | Alice Taylor | 3/07/1968 |
| richard | $2y$10$S.idSMp0yZKUZEGqmSCRMeG33IZJrtWCYvzbQc.rYVKe8fmBdq3mm | Bob Saget | 8/16/1965 |

### DML – Add Operations (20 points)

Submit a query (or queries) which **adds** the following **operation** and **consider the needed relational data**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **type** | **reason** | **sum** | **notes** | **for\_date** | **user** |
| O | Coffee | 2,50 | Costa | 10/4/2017 | peter |
| I | Salary | 500,00 |  | 7/3/2017 | Alice |
| O | Party | 120,09 | Club | 11/6/2017 | Alice |
| O | Coffee | 0,90 | Game Bar | 9/17/2017 | Alice |

### DML – Get All Users (10 points)

Get all users data, **without** the **password** and the **primary key**, ordered by their **age,** then by **primary key**:

|  |  |  |
| --- | --- | --- |
| **username** | **full\_name** | **year of born\_on** |
| James | James Cameron | 1945 |
| peter | Peter Jakson | 1952 |
| richard | Bob Saget | 1965 |
| Alice | Alice Taylor | 1968 |

### DML – Get a Range of the Operations (15 points)

Get **the second and third operations** from the table of operations, ordered by **for\_date (from past to present)**, then by the **primary key ascending**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **type** | **reason\_id** | **sum** | **user\_id** |
| 4 | O | 2 | 0.90 | 3 |
| 1 | O | 2 | 2.59 | 1 |

### DML – Get All Operations (20 points)

Get all operation’s **primary key, type, reason name, sum, user name** ordered by **primary key ascending**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **type** | **reason** | **sum** | **user name** |
| 1 | O | Coffee | 2.50 | peter |
| 2 | I | Salary | 500.00 | Alice |
| 3 | O | Party | 120.09 | Alice |
| 4 | O | Coffee | 0.90 | Alice |

## HTML

### Register Form (20 points)

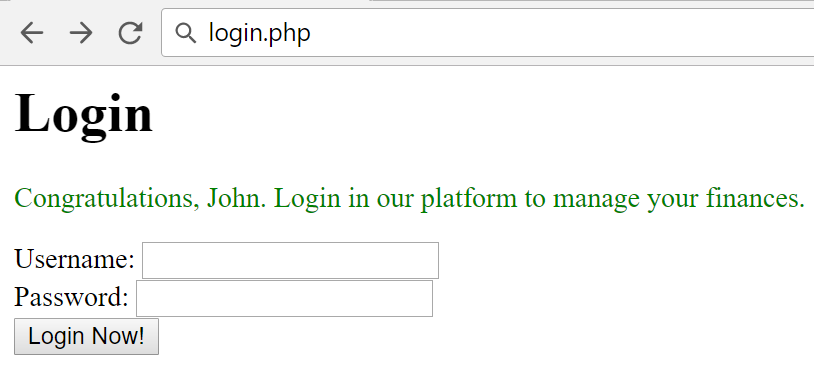
Submit a **zip archive** with an **HTML file which is a form**, registering users. Follow the constraints below:

1. **Username** with minimum length of **4** symbols
2. **Password** with minimum length of **6** symbols
3. **Confirm password** with minimum length of **6** symbols
4. **Full name** with minimum length of **5** symbols
5. **Born On** with valid date (between 1900 and current year)
6. Button, submitting the form with value “**Register now**”

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

### Successful Register Page (30 points)

Submit **zip archive** with **“index.php”** and **“login.php”** which accepts **success** message in the session and prints it above the login form as shown:



### Login Page (20 points)

Submit a **zip archive** with an **HTML file with form** with:

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

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

### New Operation Page (40 points)

Submit a **zip archive** with one html file, contains an **HTML** form with:

1. **Type** which is **dropdown** with option “Expense” and “Income”
2. **Reason** which is **empty** **dropdown** (will be later populated with PHP)
3. **Sum** with minimum value of 0 and maximum - 100 000
4. **Notes** with minimum length of **0** symbols
5. **For date** containing **valid date**
6. **Button**, submitting the form with text “**Save**”
7. **Hyperlink** to index page

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

## Structure

After each subtask below, submit a **zip archive** with the project in judge. 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 (30 points)

1. Create a file named **“export.sql”** holding all the CREATE TABLE statements (**without** CREATE DATABASE). 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 operation, edit operation, statistics** and **logout**;
4. Extract common operation such as autoloading classes in **“common.php”.**

### Database Abstraction Layer (50 points)

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 (60 points)

Following the naming convention **“{EntityName}Repository”**, create repository layer in the namespace **\MyMoney\Repository**

### Service Layer (60 points)

Following the naming convention **“{BusinessEntity}Service”,** create service layer in the namespace **\MyMoney\Service**

## MyMoney Management Application (200 points)

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