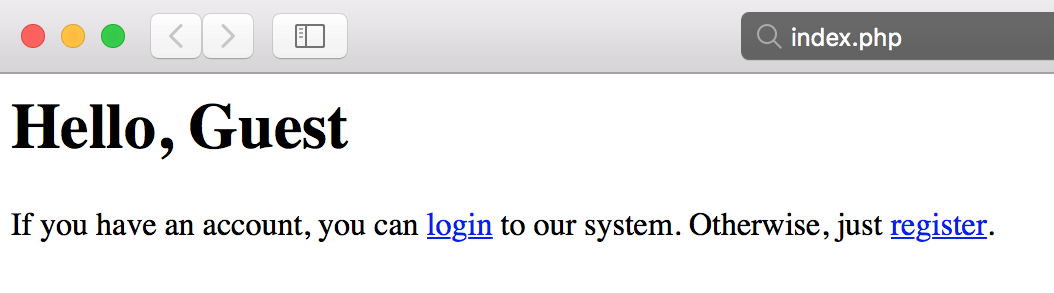
# Exam Preparation – Task Management System

You need to create a **web application** where users **see and manage list of tasks**. Usually the users will visit the page, navigate either to **signing up or in**, or going to the **dashboard**, where all **tasks are listed**. They can either **add** a new task, **edit** or **delete** existing, **or see the details** of an existing task.

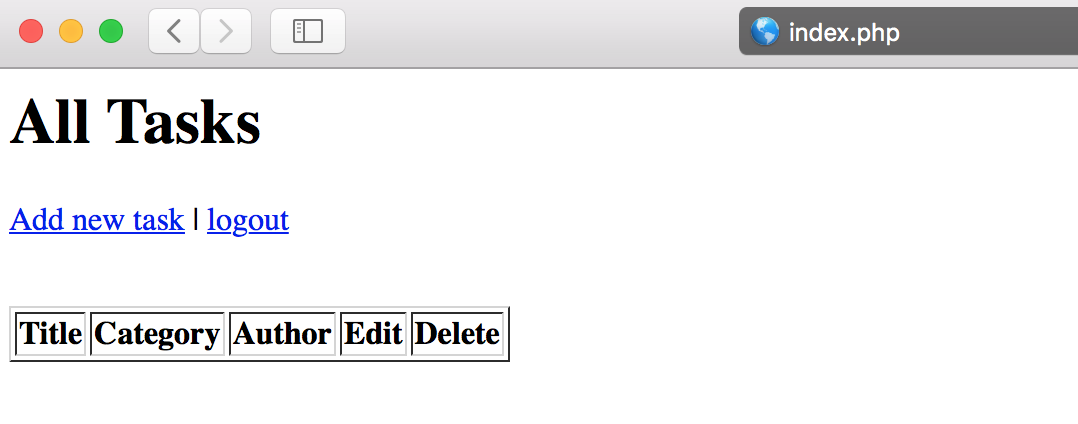
## Expected Result

index.php:

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

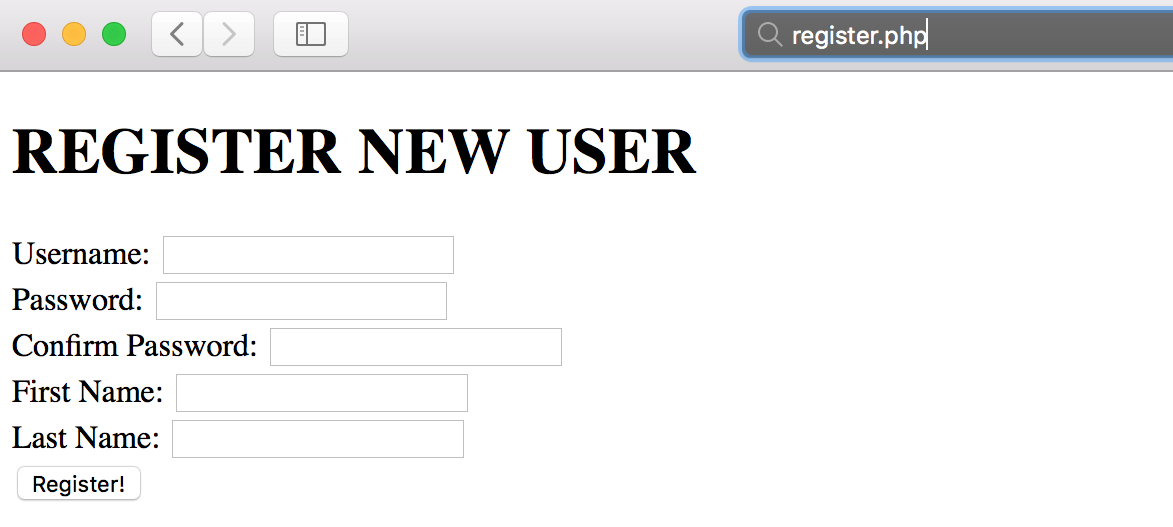


* When user **is** logged:

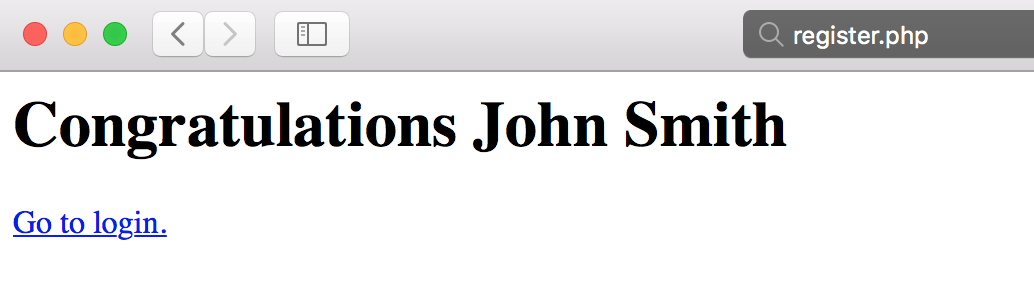


register.php:

* **Before** form is submitted:

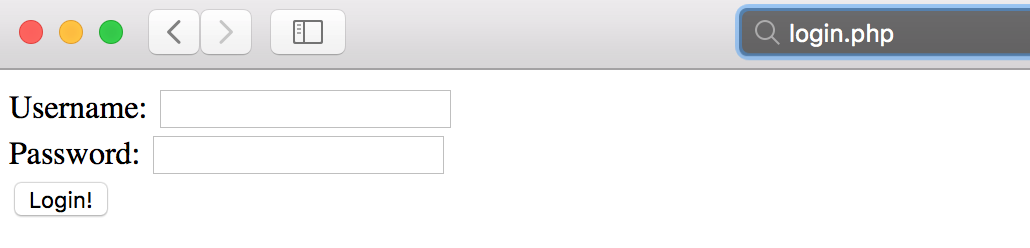


* **After** successful submission:

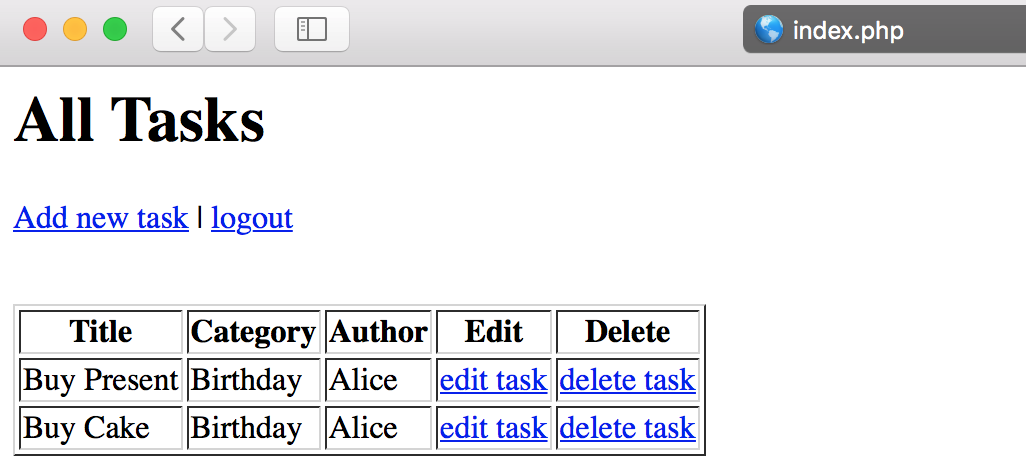


login.php:

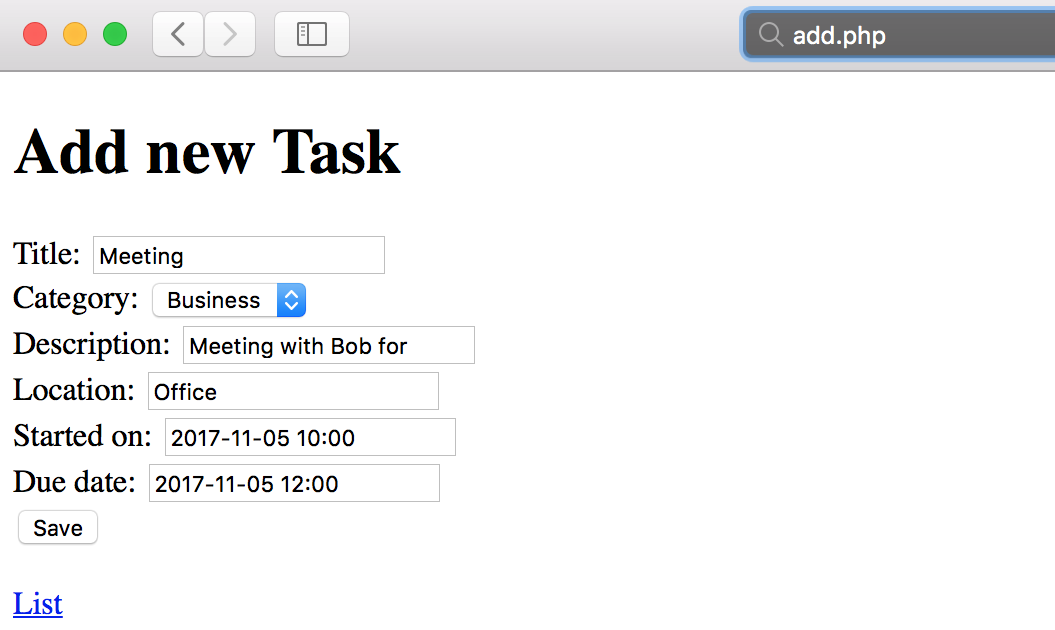
* **Before** form is submitted:



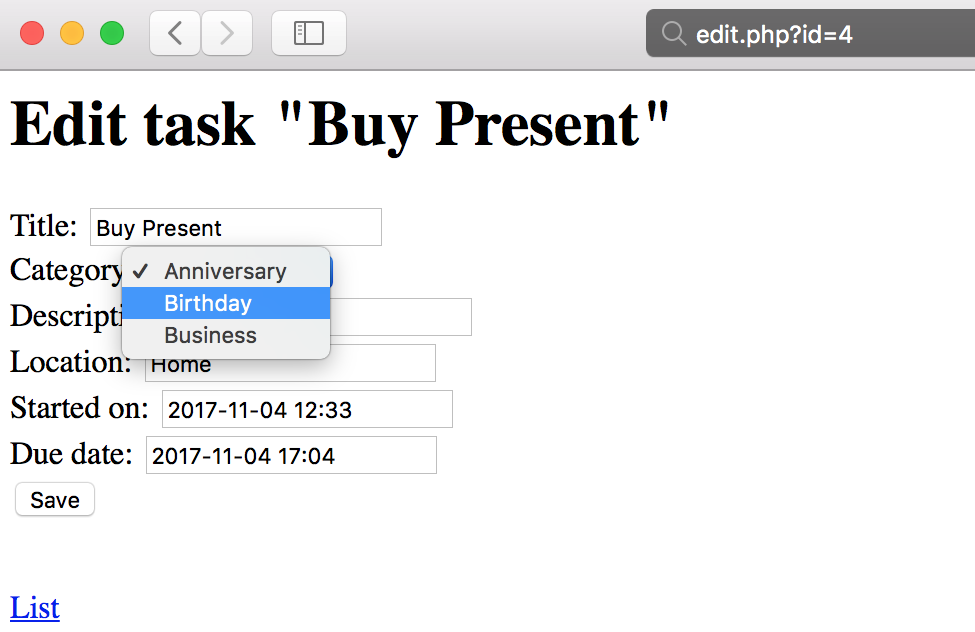
* **After** successful login (task list ordered by the sooner **due date**, then by **id**):



add.php (Add new task):



edit.php?id={task\_id} (Edit existing task):



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

**Deletes** the task and **redirects** to task list again.

## Database

Create a database “exam\_prep” and add the following tables:

### Data Definition

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

### DDL – Users

|  |  |  |
| --- | --- | --- |
| **users** | | |
| **Column Name** | **Type** | **Constraints** |
| username | String | 4…255. Unique. |
| password | String | 6…255 |
| first\_name | String | 3…255 |
| last\_name | String | 3…255 |

### DDL – Tasks

|  |  |  |
| --- | --- | --- |
| **tasks** | | |
| **Column Name** | **Type** | **Constraints** |
| title | String | 3…50 |
| description | String | 10…10,000 |
| location | String | 3…100 |
| author\_id | int | 1...4294967295 |
| category\_id | int | 1...4294967295 |
| started\_on | Date Time | Possible NULL |
| due\_date | Date Time | Possible NULL |

### DDL – Categories

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

### DDL – Relations

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 Categories

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

|  |
| --- |
| **name** |
| Anniversary |
| Birthday |
| Business |

## HTML

### Register Form

Submit a **zip archive** with an **HTML 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. First name with minimum length of **3** symbols
5. Last name with minimum length of **3** symbols
6. Button, submitting the form with value “**Register!**”

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

### Successful Register Page

Submit **zip archive** with blank “index.php” and “success.php” with:

1. **Heading 1** text: “**Congratulations {username}**”, where {username} is taken from the session with the same key
2. **Hyperlink** to the login page with text “**Go to login.**”

### Login Page

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

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 Task Page

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

1. Title with minimum length of **3** symbols
2. Description containing **10 to 10,000** symbols
3. Category which is **empty** **dropdown** (will be later populated with PHP)
4. Location with minimum length of **3** symbols
5. Started on containing **valid date and time**
6. Due date containing **valid date and time**
7. Button, submitting the form with text “**Add!**”

All fields are **required**. 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

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, success, login, dashboard, add task, edit task 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

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

### Data Layer

Following the naming convention “{EntityName}DTO”, create data layer in the namespace \TaskManagement\DTO

### Service Layer

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

### Template Layer

Create Templates folder in the namespace \TaskManagement\Temlates. All templates should be in the \Templates namespaces.

### Http Layer

Following the naming convention “{EntityName}Http” - Handler, create http layer in the namespace \TaskManagement\Http

## Task Management Application

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